Atelier Data Science

Deep learning practice 2

Convolutional Neural Networks

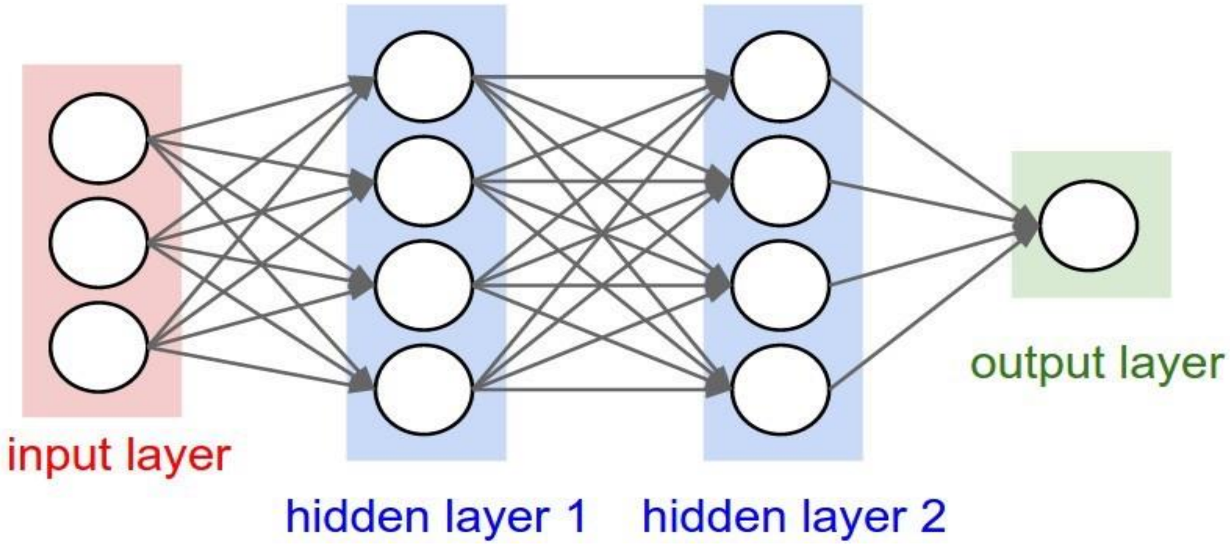
Irina Proskurina

Irina.Proskurina@univ-lyon2.fr

Laboratoire ERIC – Université Lyon 2

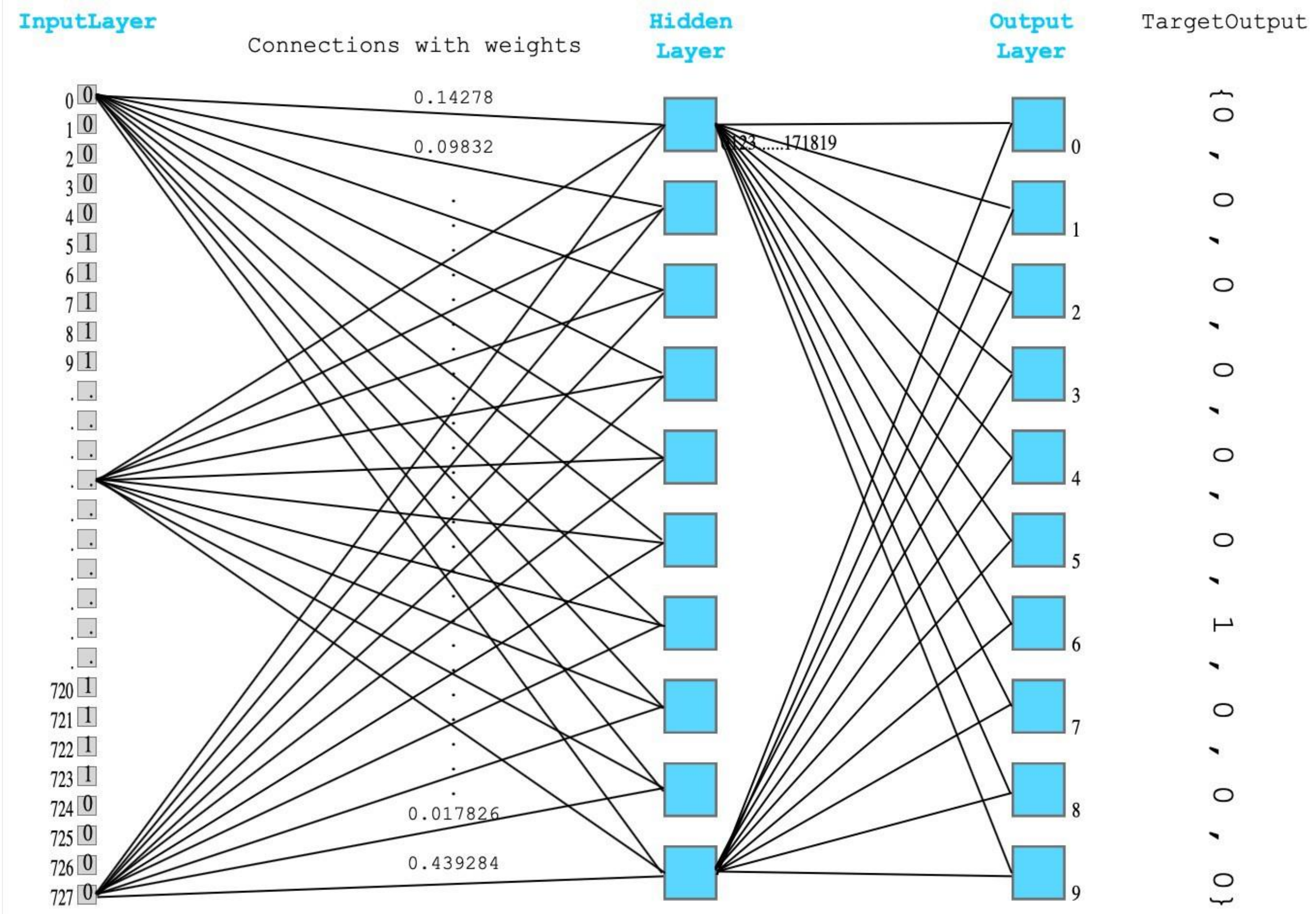
Previous Lesson Recap 1

1. Neural Networks -> no feature engineering needed
2. Fully connected layers/Dense/nn.Linear() layers
3. Fully connected neural networks : connect every neuron in one layer to every neuron in the other layer



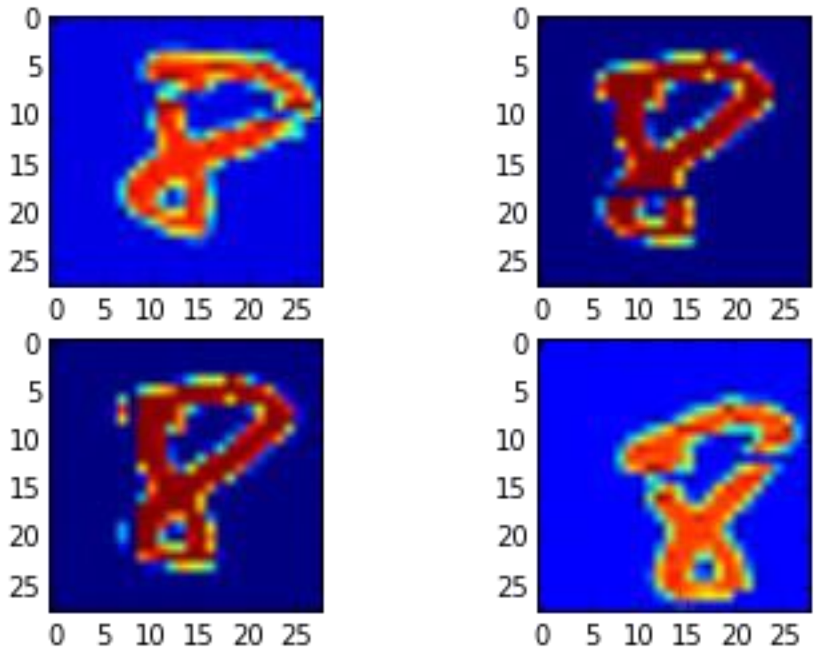
Previous Lesson Recap 2
MNIST

- Each neuron can detect the presence of a specific set of pixels



Previous Lesson Recap 2
MNIST

- If you shift the digit slightly, the neuron will no longer detect its pattern



https://srome.github.io/Jitter,-Convolutional-Neural-Networks,-and-a-Kaggle-Framework/

Number of parameters

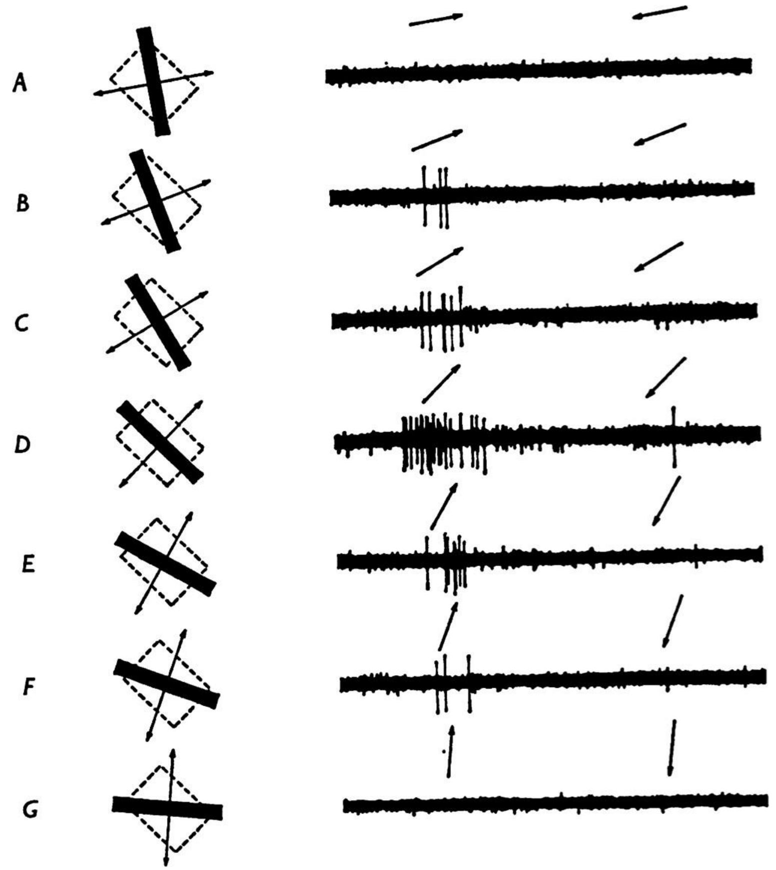
- 784 inputs
- Fully connected layer: 1000 neurons
- Output layer: 10 neurons (one for each class)
- Weights between input and fully connected layers: (784 + 1) * 1000 = 785,000
- Weights between fully connected and output layers: (1000 + 1) * 10 = 10,010

Fully connected neural networks for image classification

- A lot of parameters
- Prone to overfitting
- Does not consider the specifics of images (shifts, slight changes in shape, etc.)
- One of the best ways to combat overfitting is to reduce the number of parameters

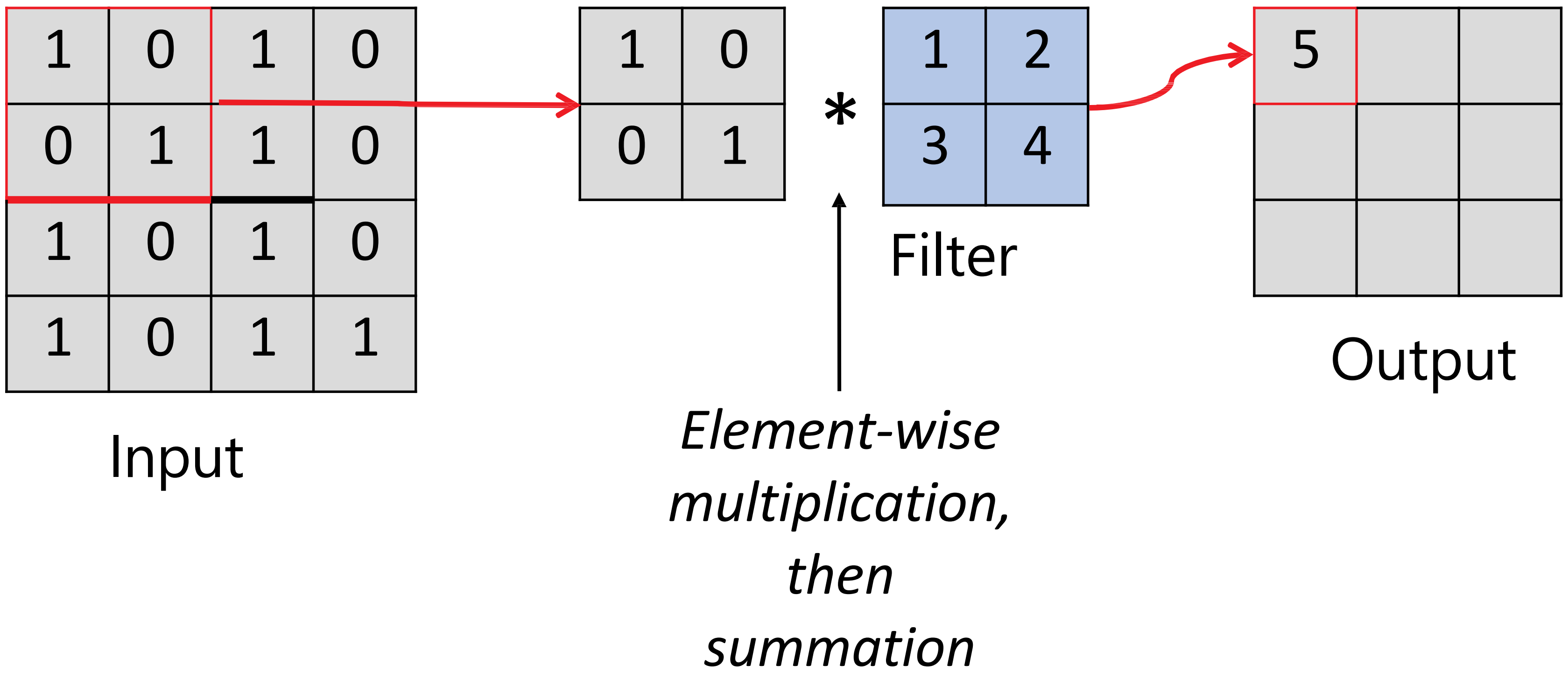
Convolutional neural network

Experiments with the visual cortex



https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1557912/

Convolution



Convolution

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{2}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{2}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{1}$$

$$\begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{0}$$

$$\begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{6}$$

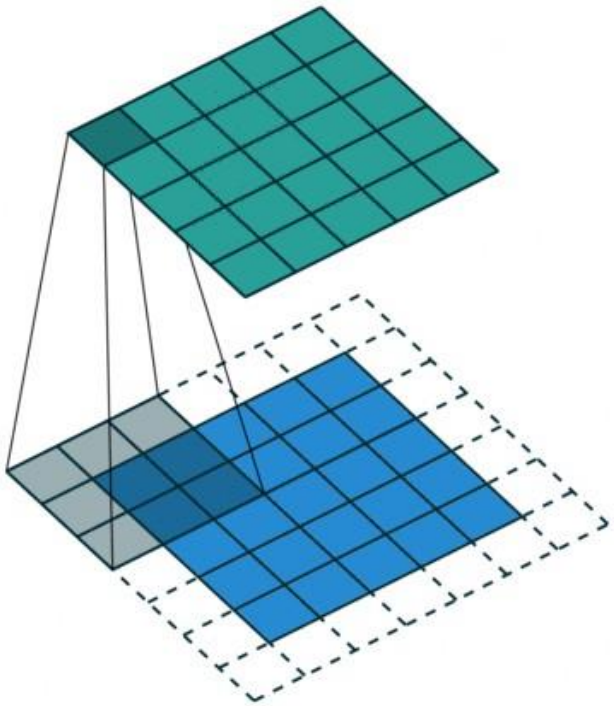
$$\begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{10}$$

Convolution

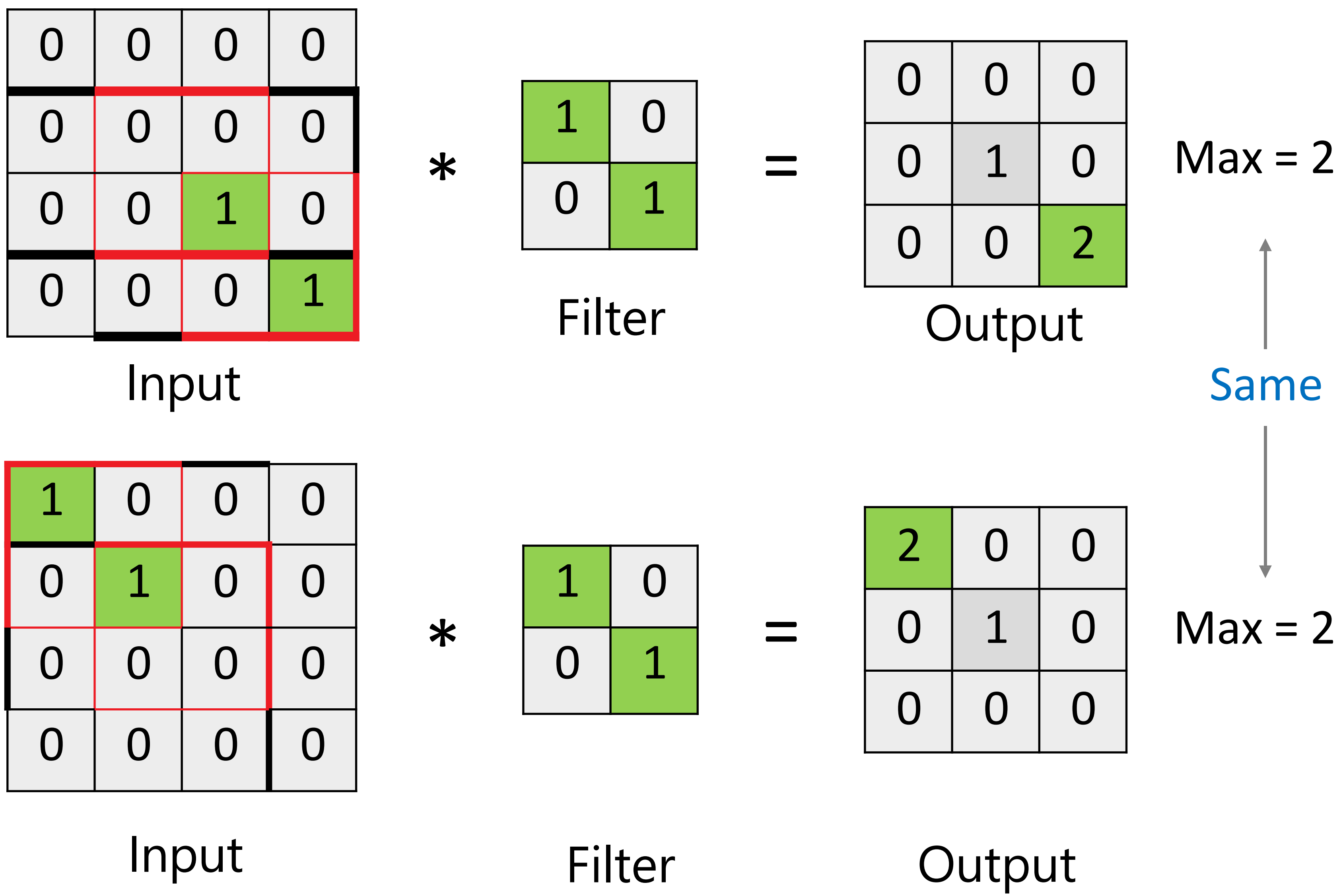
- Detects a pattern in the image, which is defined by a filter
- The stronger the pattern is represented in a particular area of the image, the higher the convolution value will be

Convolution

- The result of convolving an image with a filter is a new image



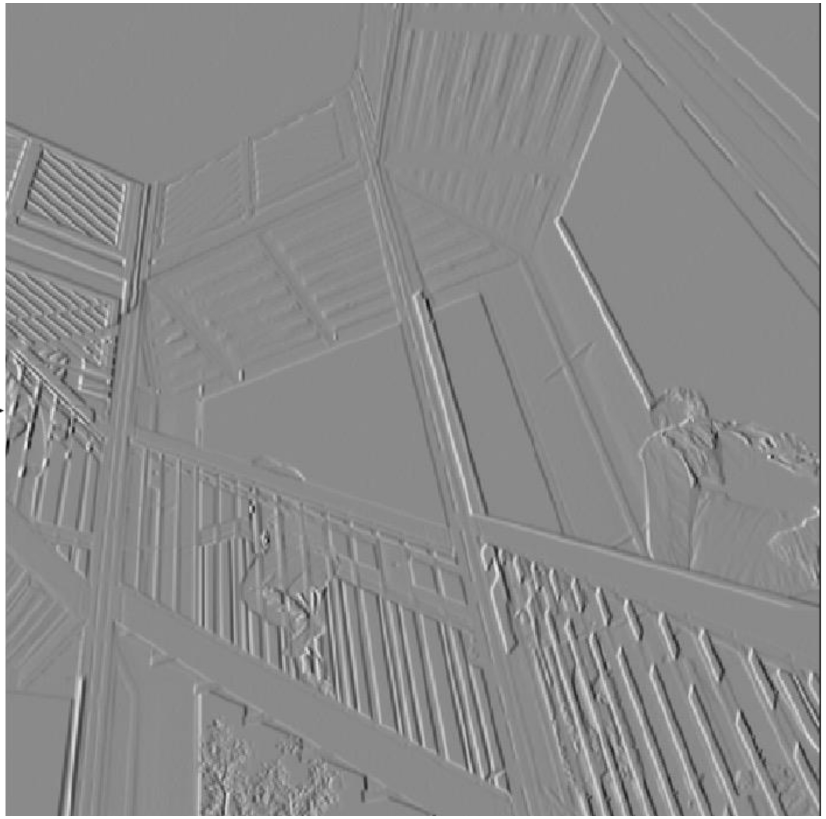
The convolution maximum is invariant to shifts



Convolutions in computer vision

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Horizontal Sobel kernel



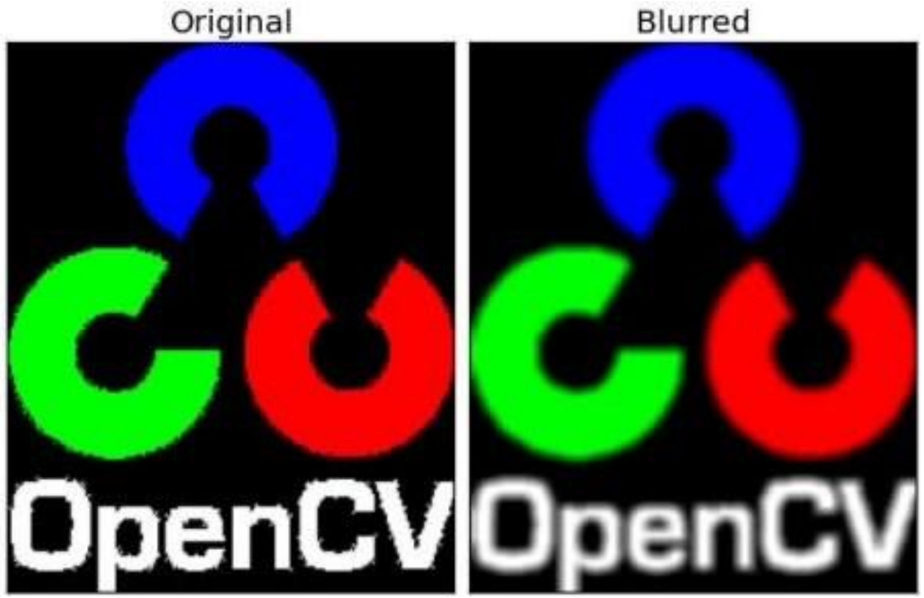
https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Convolutions in computer vision

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

Convolutions in computer vision



$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html

Convolution

$$\mathrm{Im}^{out}(x,y) = \sum_{i=-d}^{d} \sum_{j=-d}^{d} \big(K(i,j)\,\mathrm{Im}^{in}(x+i, y+j) + b\big)$$

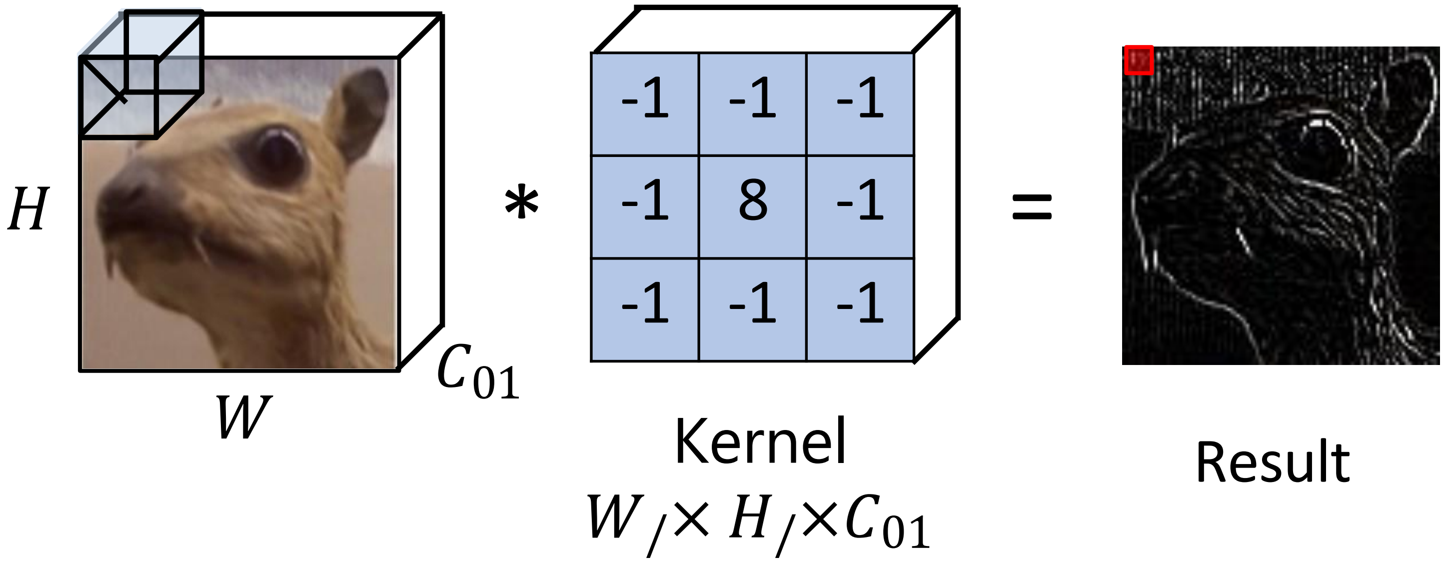
- A pixel in the resulting image depends only on a small region of the input image (local connectivity)
- The weights are the same for all pixels in the output image (shared weights)

Convolution

- Usually, the original image is colored!
- This means that it has multiple channels (R, G, B). Let's consider it in the formula:

$$\mathrm{Im}^{out}(x,y) = \sum_{i=-d}^{d} \sum_{j=-d}^{d} \sum_{c=1}^{C} \big(K(i,j,c)\,\mathrm{Im}^{in}(x+i,y+j,c) + b\big)$$

Convolution



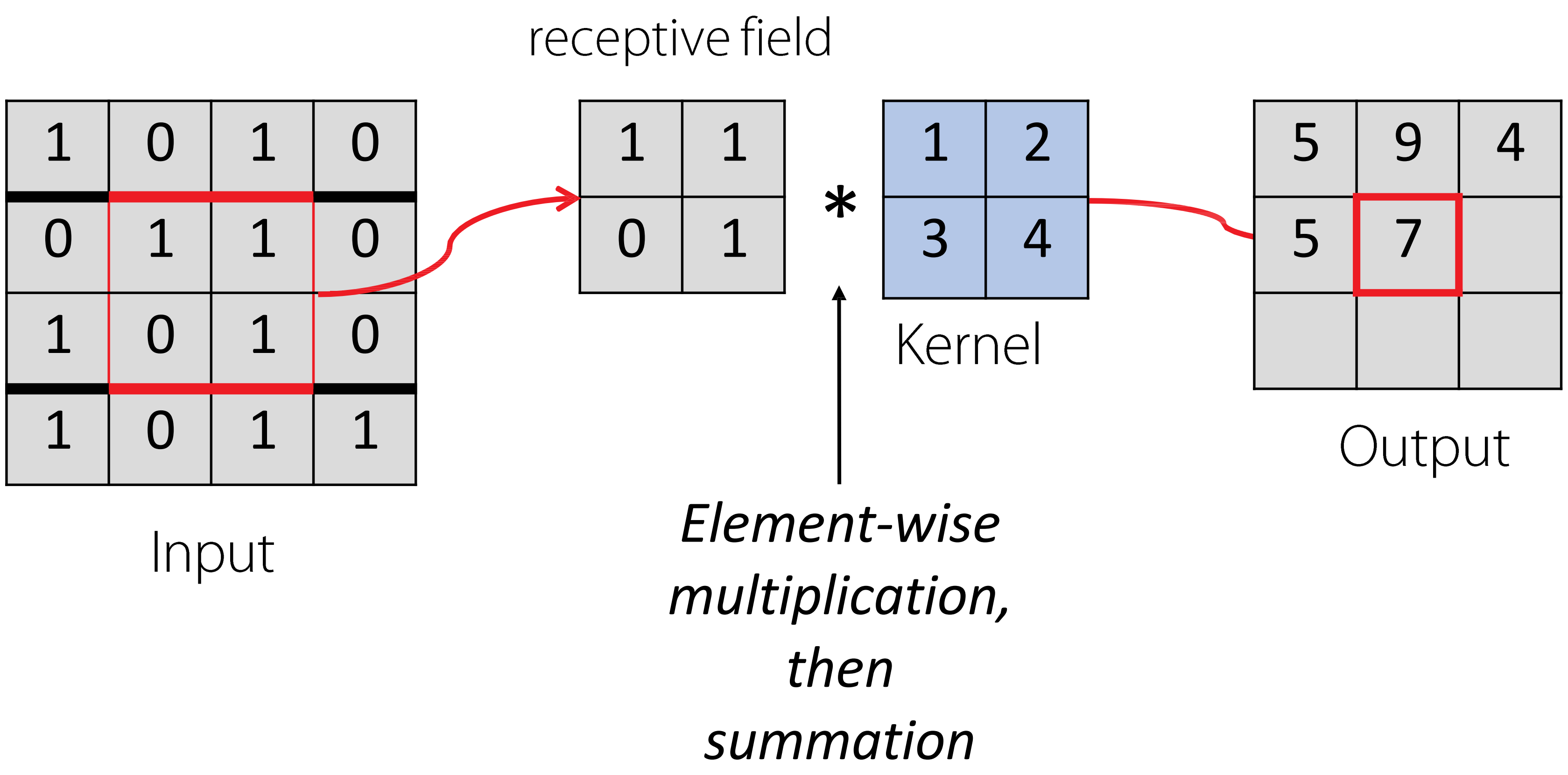
Number of parameters

$$\mathrm{Im}^{out}(x,y,t) = \sum_{i=-d}^{d} \sum_{j=-d}^{d} \sum_{c=1}^{C} \left(K_t(i,j,c)\, \mathrm{Im}^{in}(x+i, y+j, c) + b_t \right)$$

- Kernel parameters
- $\left((2d+1)^2 * C + 1\right) * T$

Receptive field

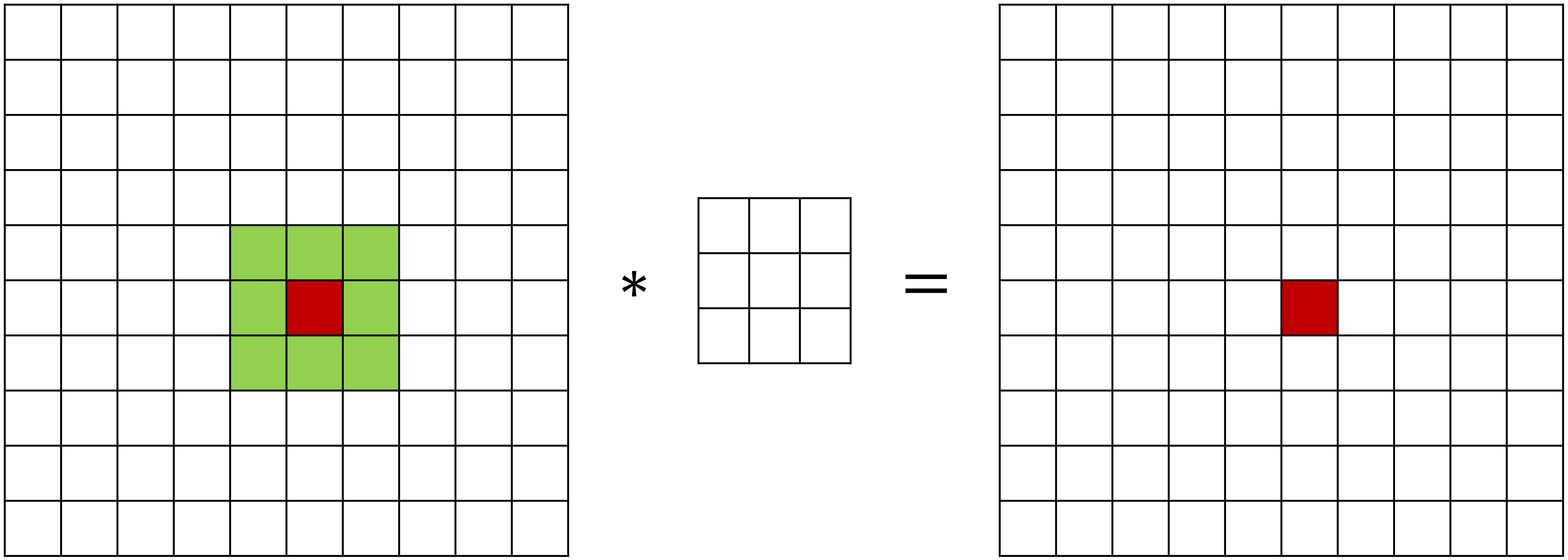
Convolution



Receptive field

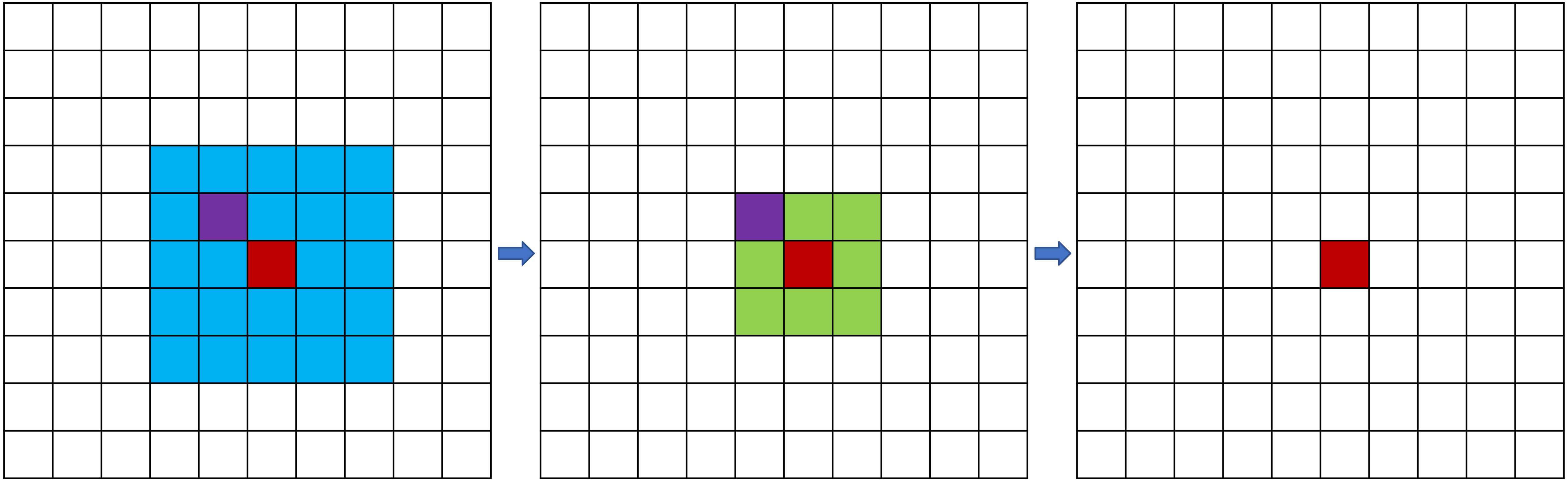
- Let's take a pixel in the output image
- Which part of the input image does the value in this pixel depend on?

Receptive field



Receptive field: 3 x 3

Receptive field



Receptive field: 5 x 5

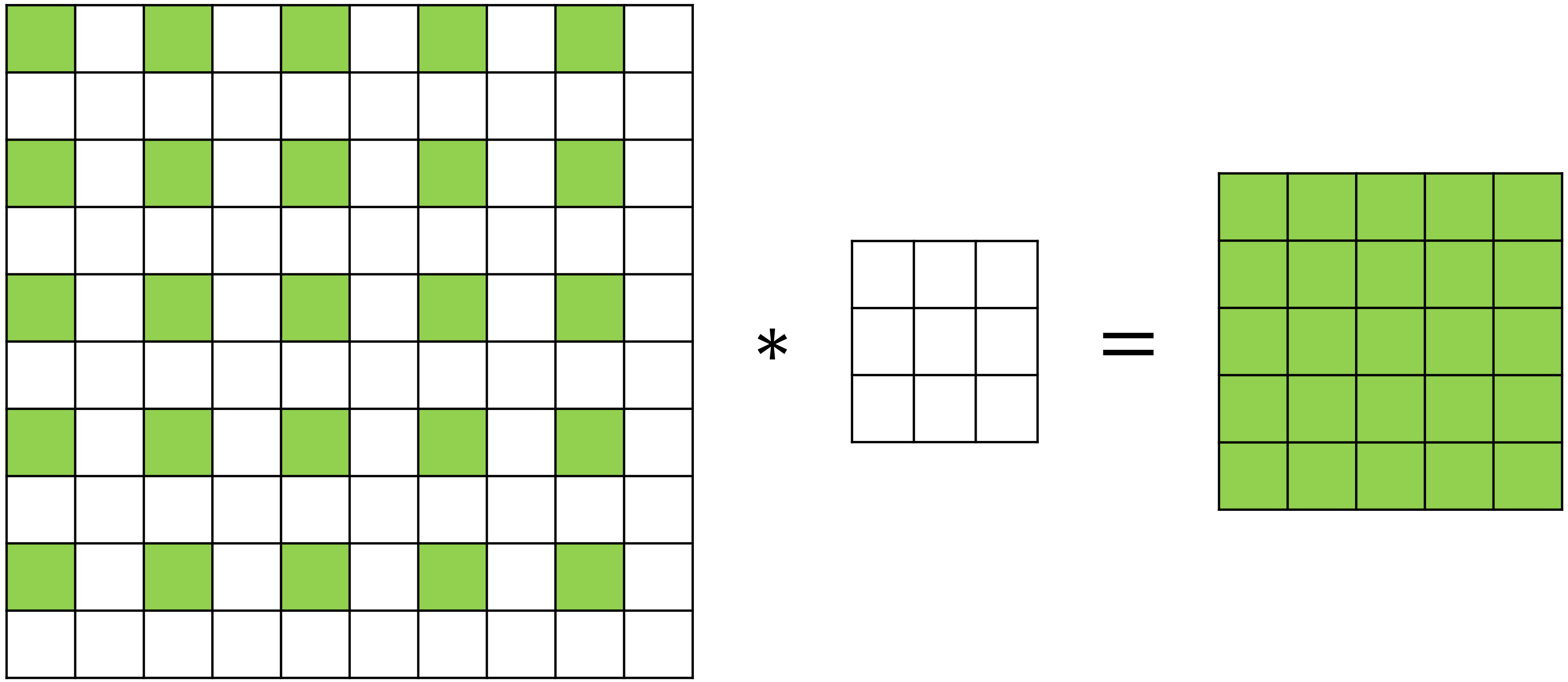
Receptive field

Receptive field for 3 x 3 convolution:

- After 1 convolutional layer: 3 x 3
- After 2 convolutional layers: 5 x 5
- After 3 convolutional layers: 7 x 7

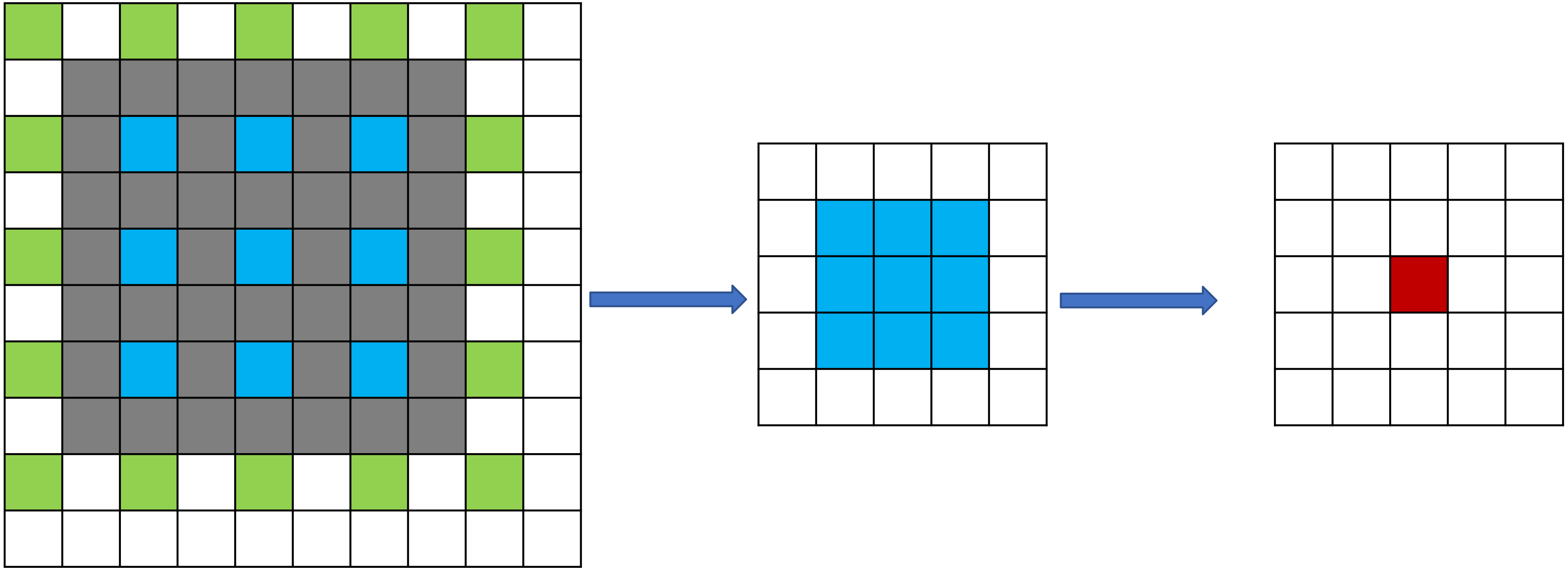
We need a lot layers if the image size is 512 x 512

Strides



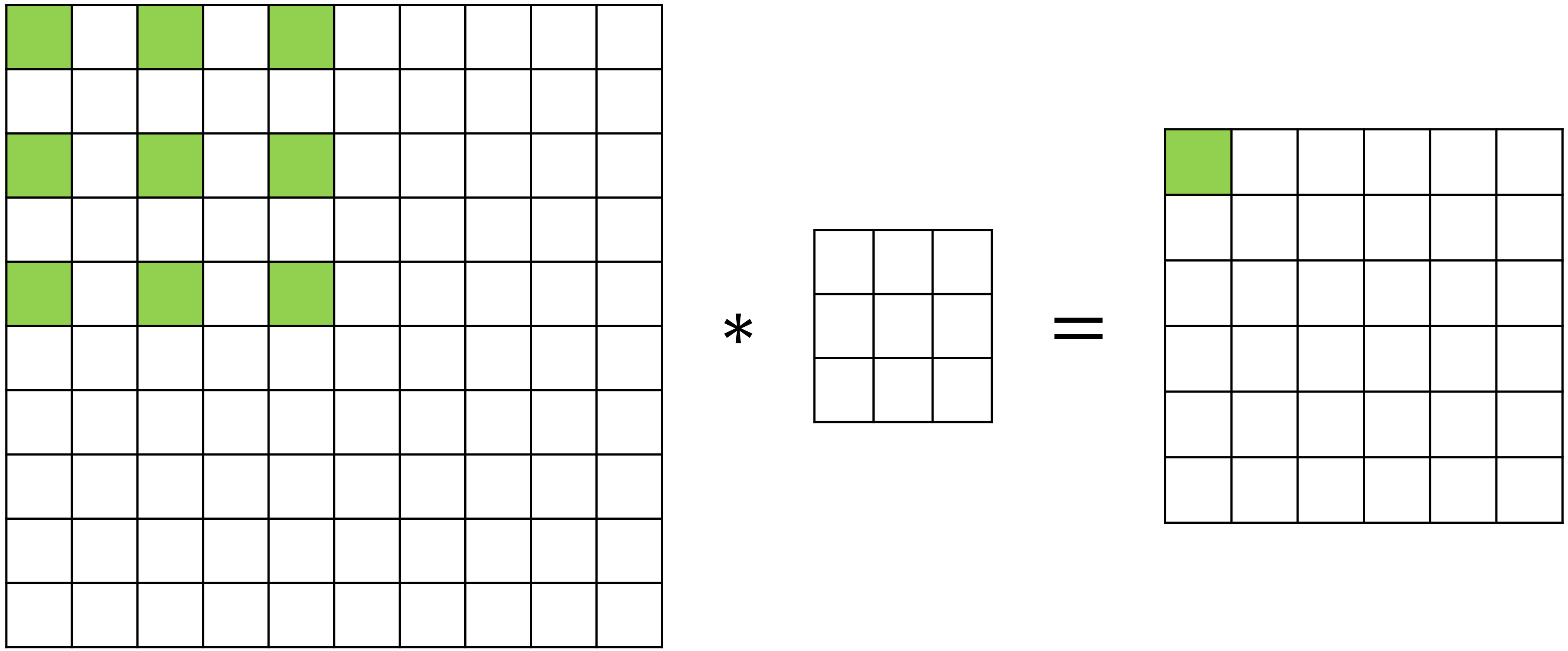
$$s = 2$$

Strides



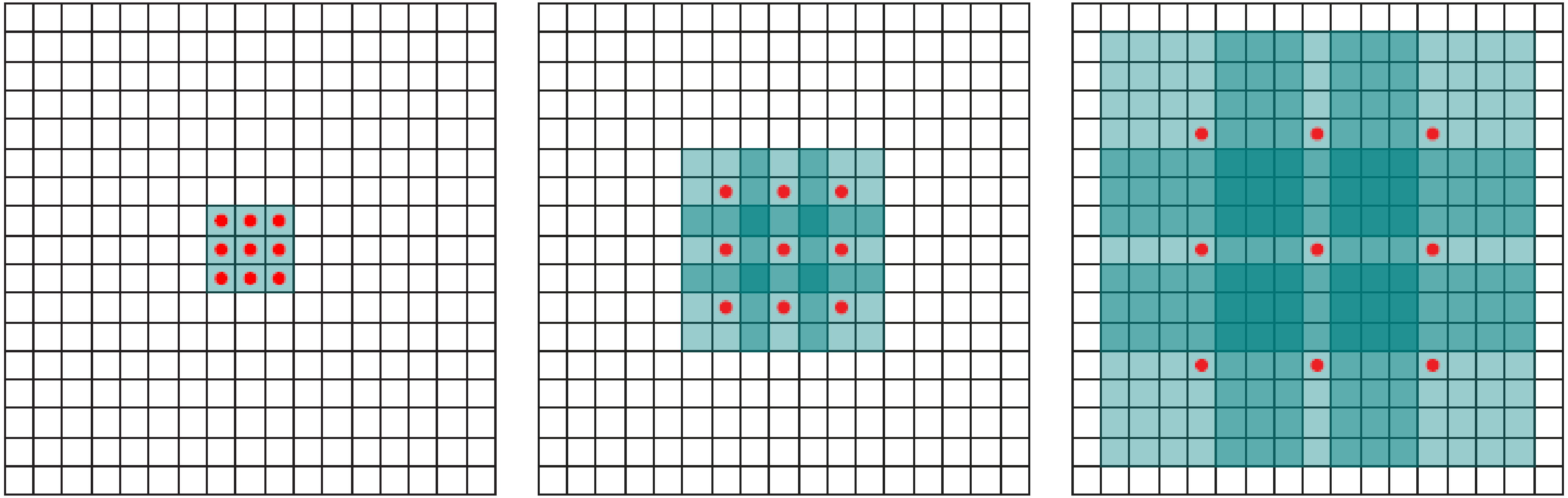
Receptive field: 7 x 7

Dilated convolutions



$$l = 2$$

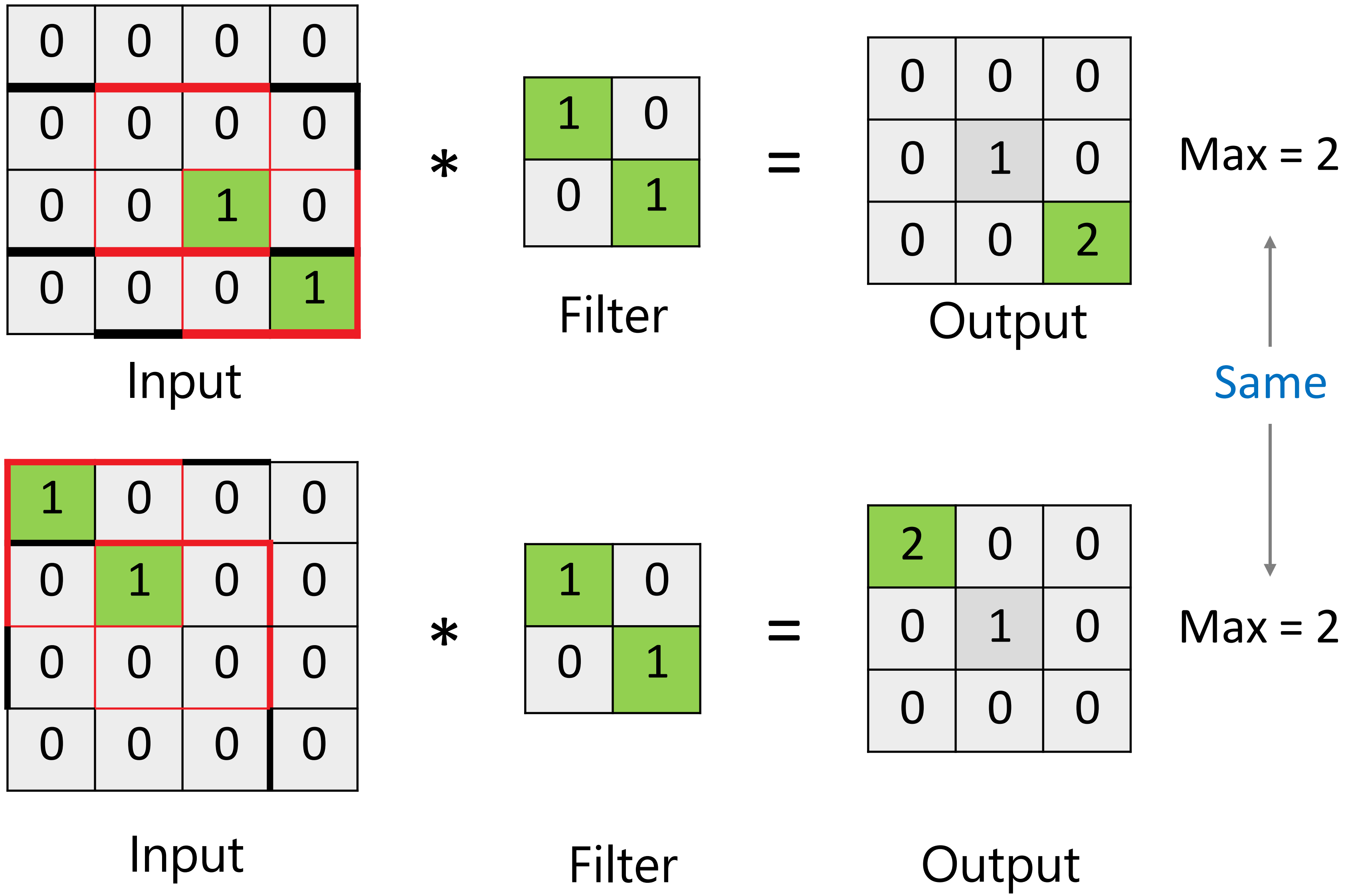
Dilated convolutions



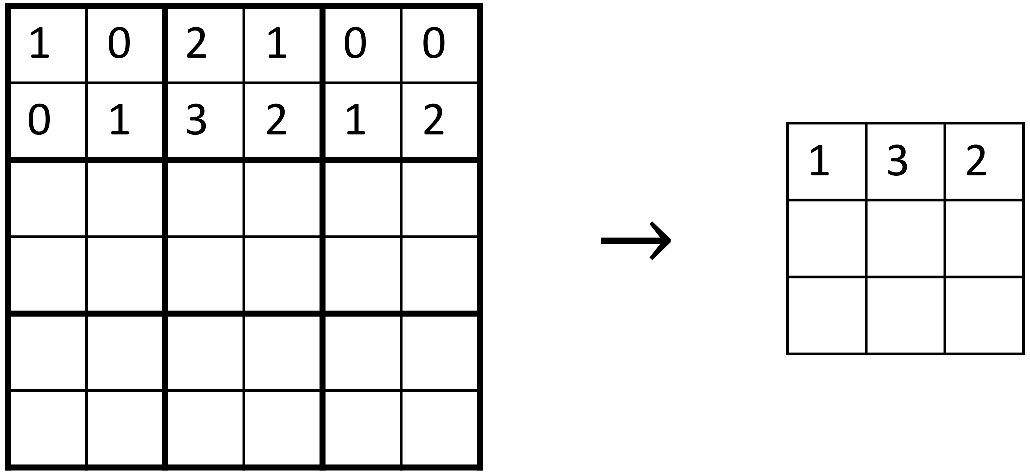
$l = 1$ $l = 2$ $l = 4$

https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5

The convolution maximum is invariant to shifts



Pooling



Max-pooling with kernel 2x2

Pooling

- Splits the image into $n \times m$ sections applying some function (usually a maximum)
- Significantly reduces the size of the image (which means it increases the field of perception of the following layers)
- Has no parameters

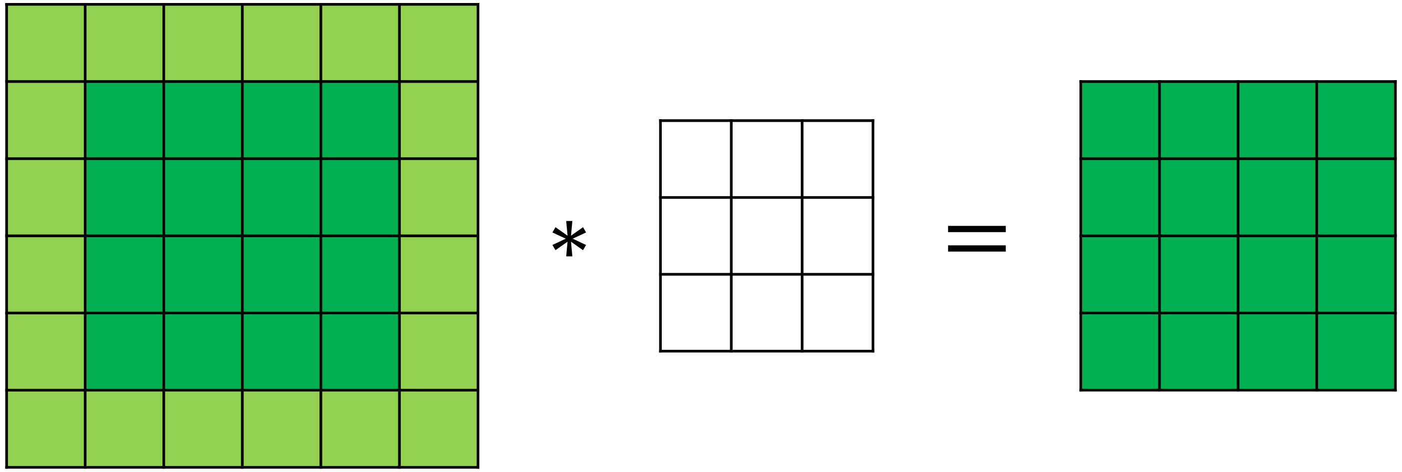
Why we need to know all this?

- It is important to ensure that the last convolutional layers have a perceptual field size comparable to the entire image

Convolution

- If you apply convolution, the output image will be smaller than the input

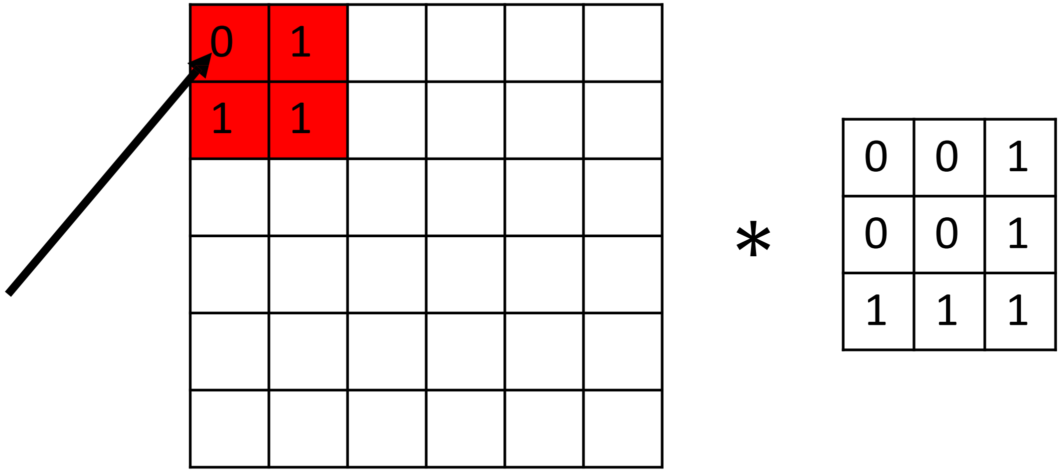
Convolutions



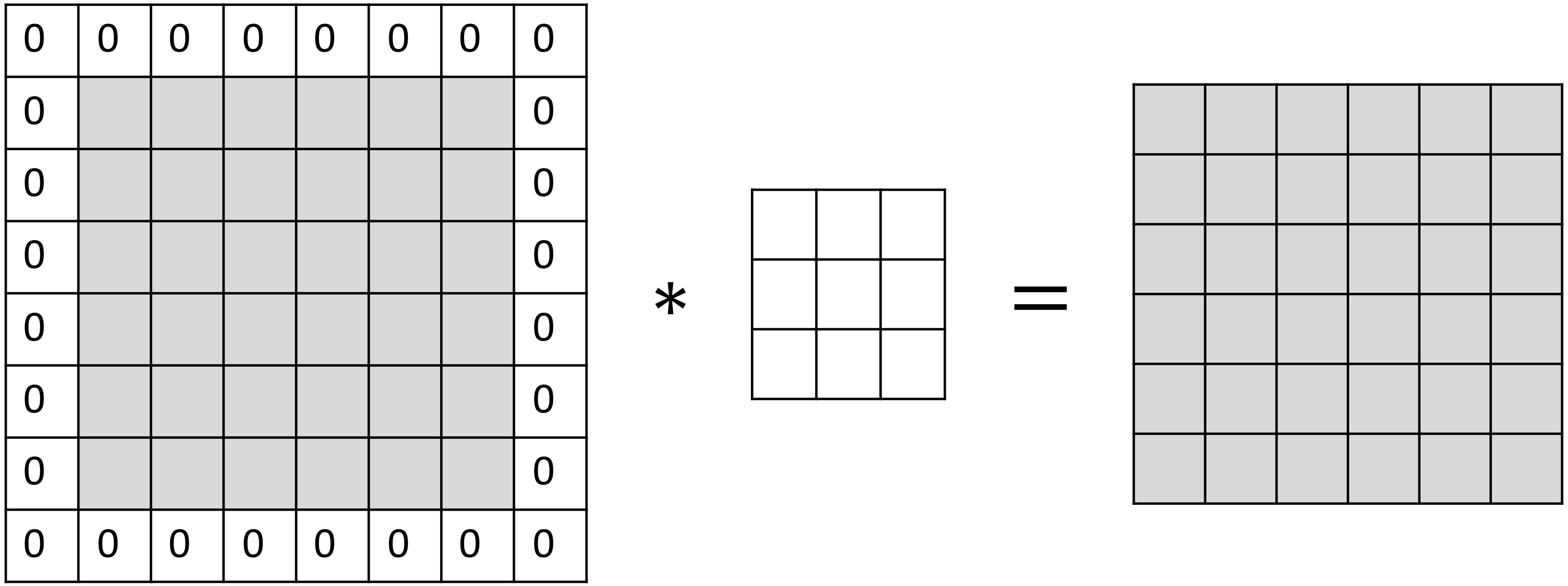
Valid mode

- When counting convolutions, the pixels at the edges do not have a big impact on the result

We will not see that the filter has a good response when placing the center at this pixel



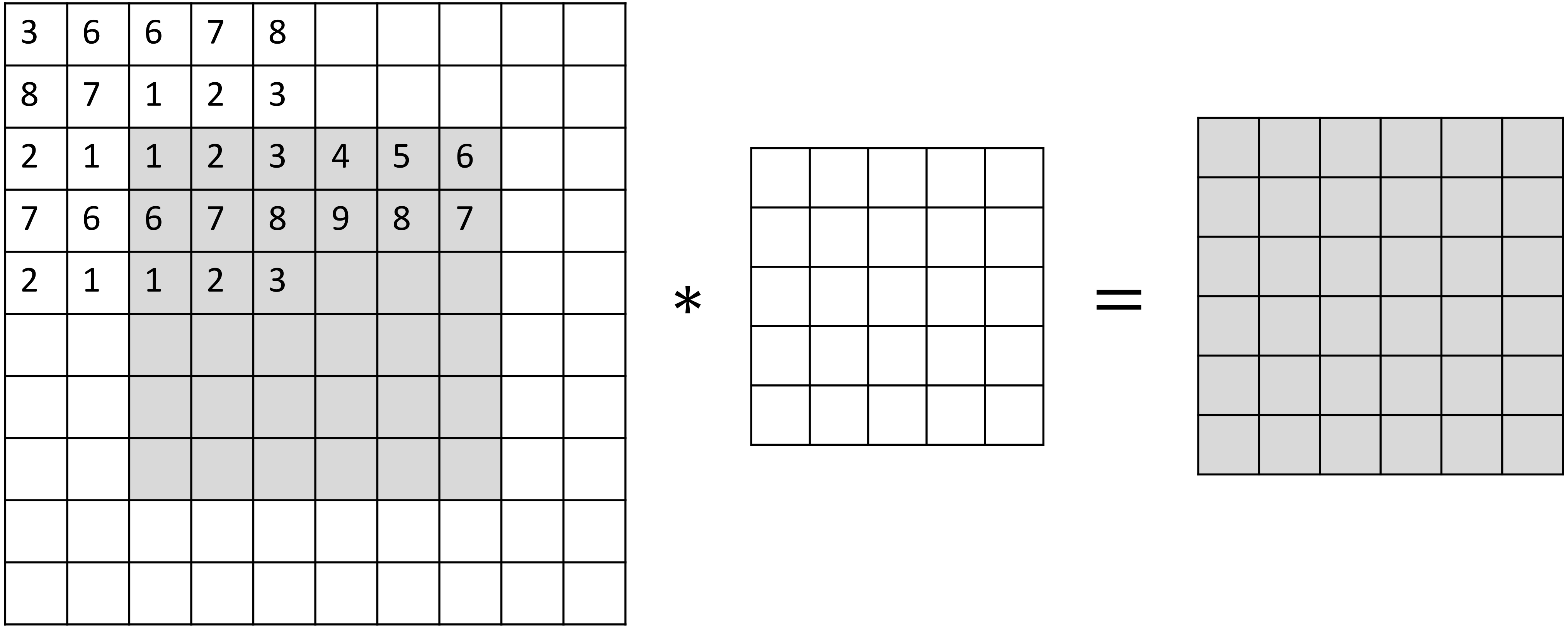
Zero padding



Zero padding

- We add zeros along the boundaries so that the convolution calculated after this in valid mode gives an image of the same size as the original one
- There is a risk that the model will learn to understand where the edges are in the image - we may lose invariance

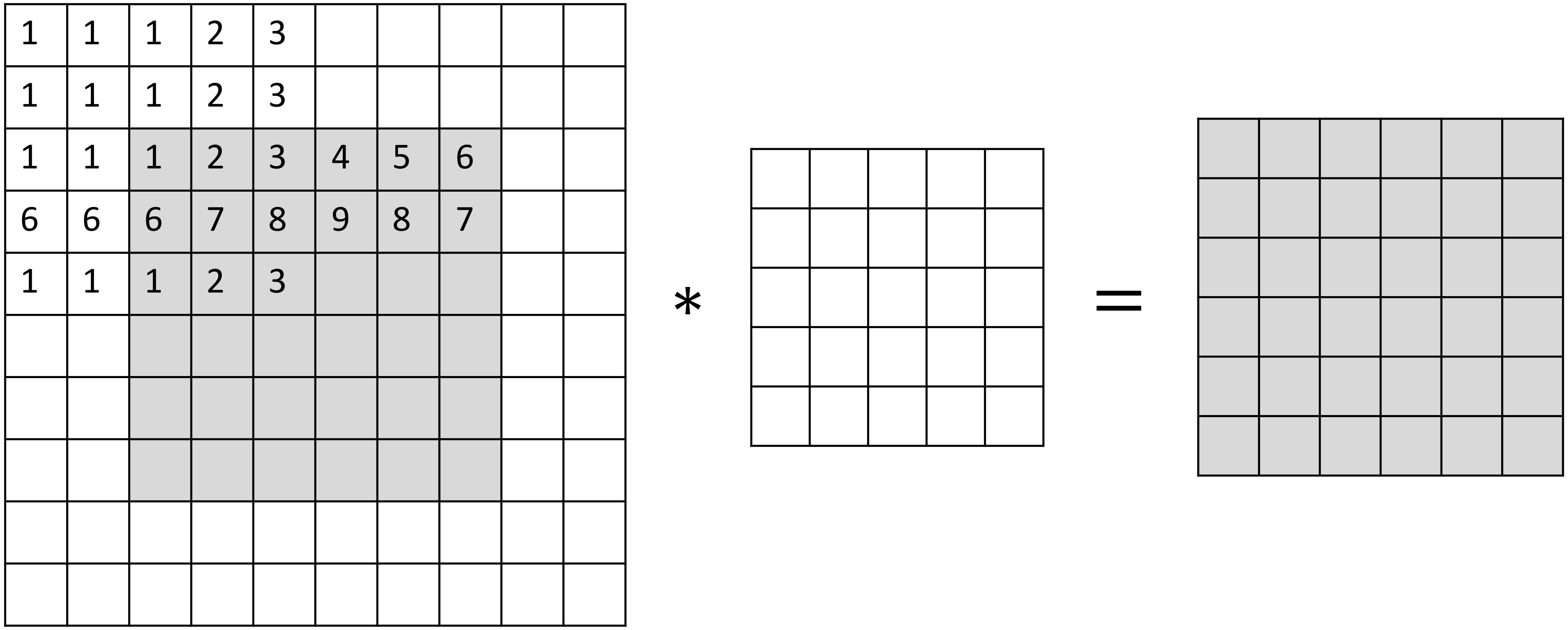
Reflection padding



Reflection padding

- Can't easily find image edges
- But now the model can begin to find specular reflections and select filters for them

Replication padding



Replication padding

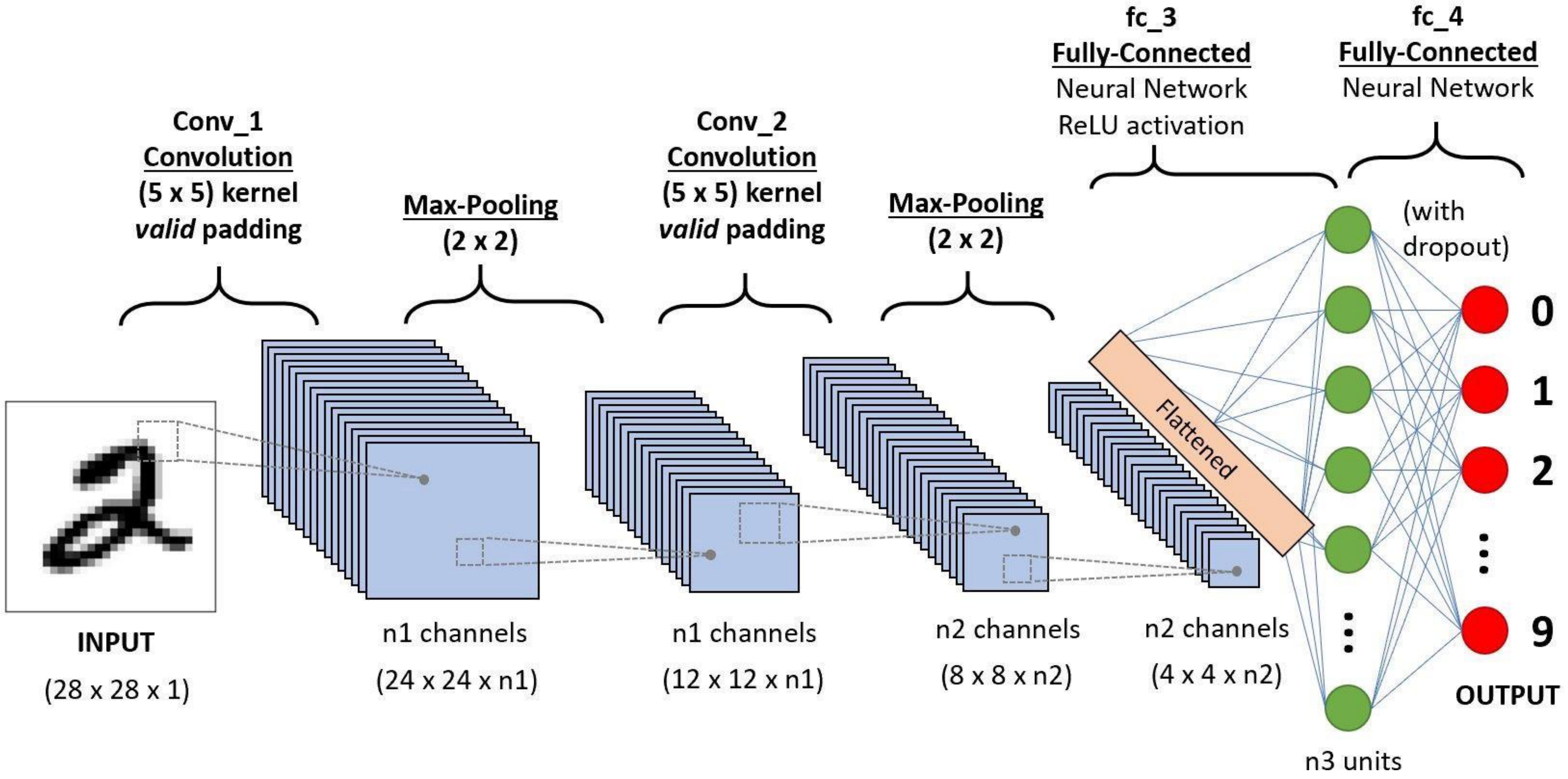
- The pixel on the border is equal to the nearest pixel from the image
- The model can still adjust to the patterns that arise from such padding

Summary

- Padding allows you to control the size of the output images
- Padding allows you to take into account objects on the edges
- Different types of padding allow different methods of retraining for edges

Convolutional Neural Networks

Architecture

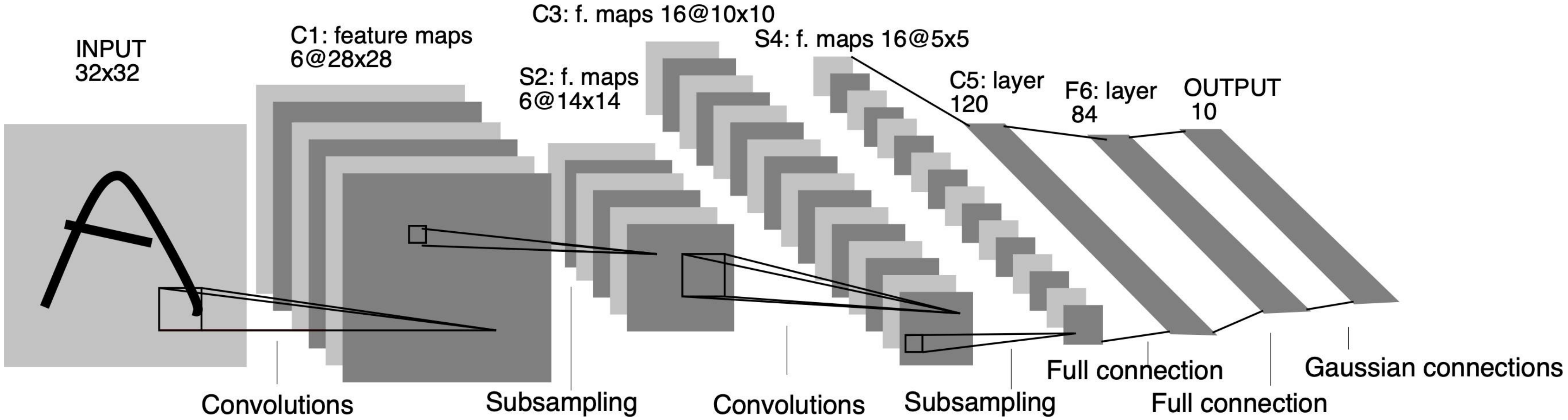


https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Architecture

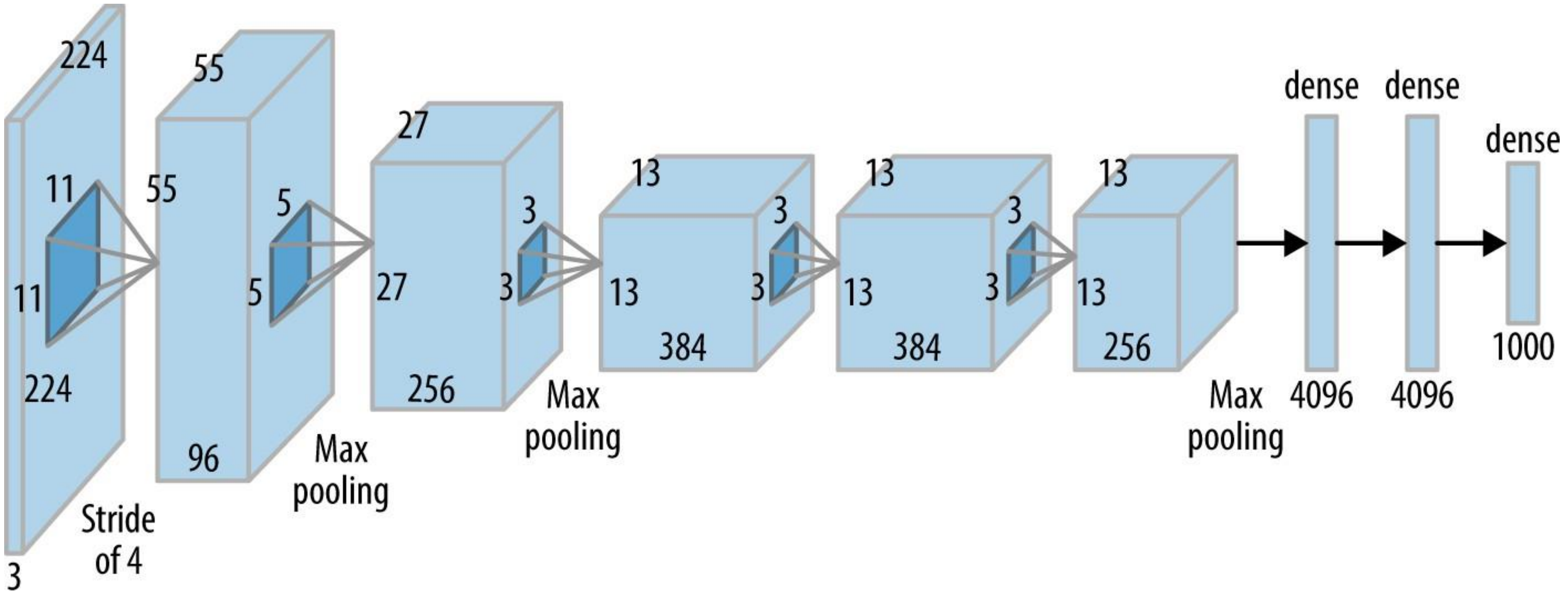
- Convolution->linear layer>pooling or convolution->non-linear layer
- flattening of the output
- Fully-connected layer

LeNet



http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

AlexNet

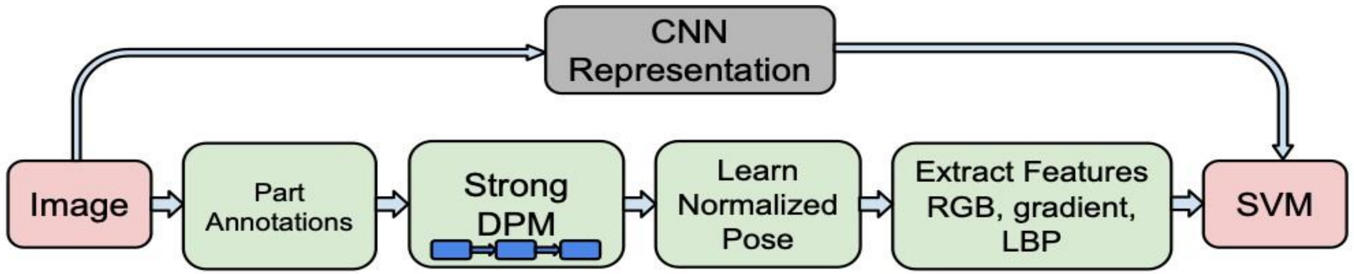


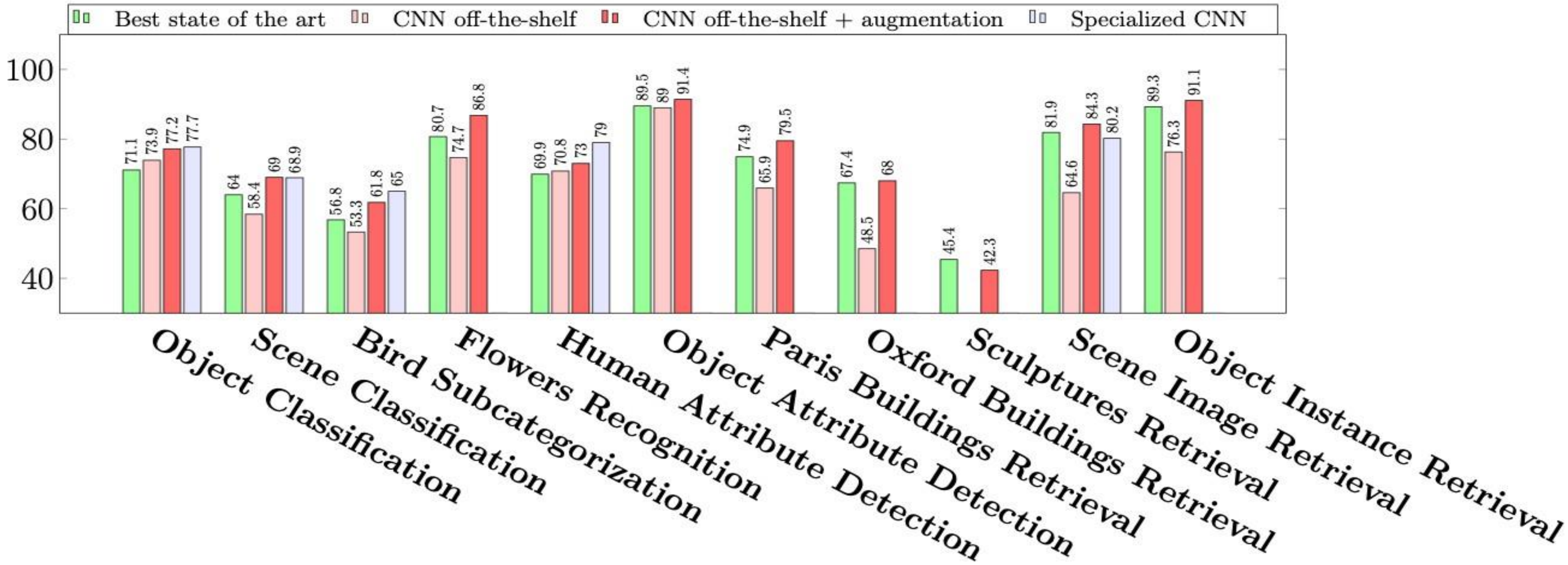
http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Image representation (embedding) from the last layers

- Important observation: the outputs of fully connected layers serve as good feature representations of images and are valuable in many tasks
- For instance, they can be utilized in tasks like searching for similar images

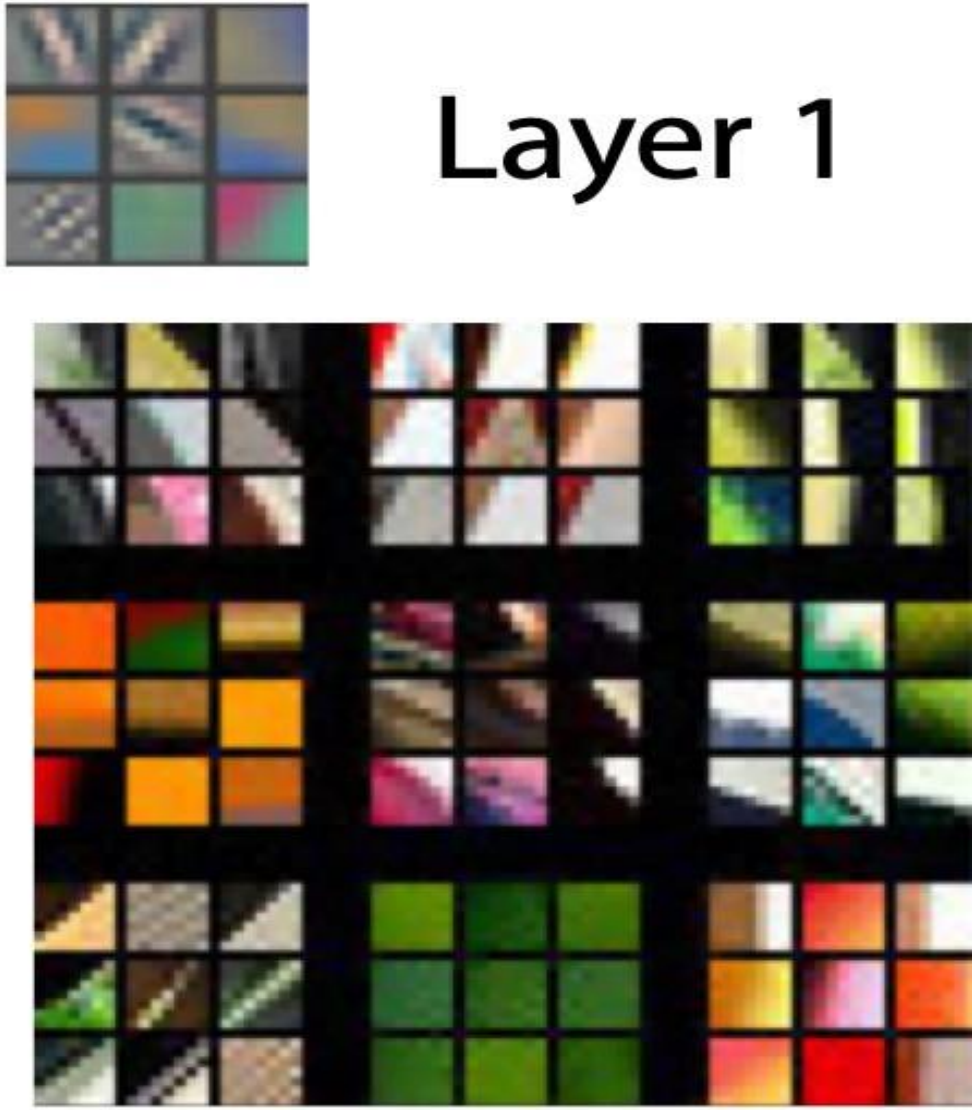
Last layer embeddings





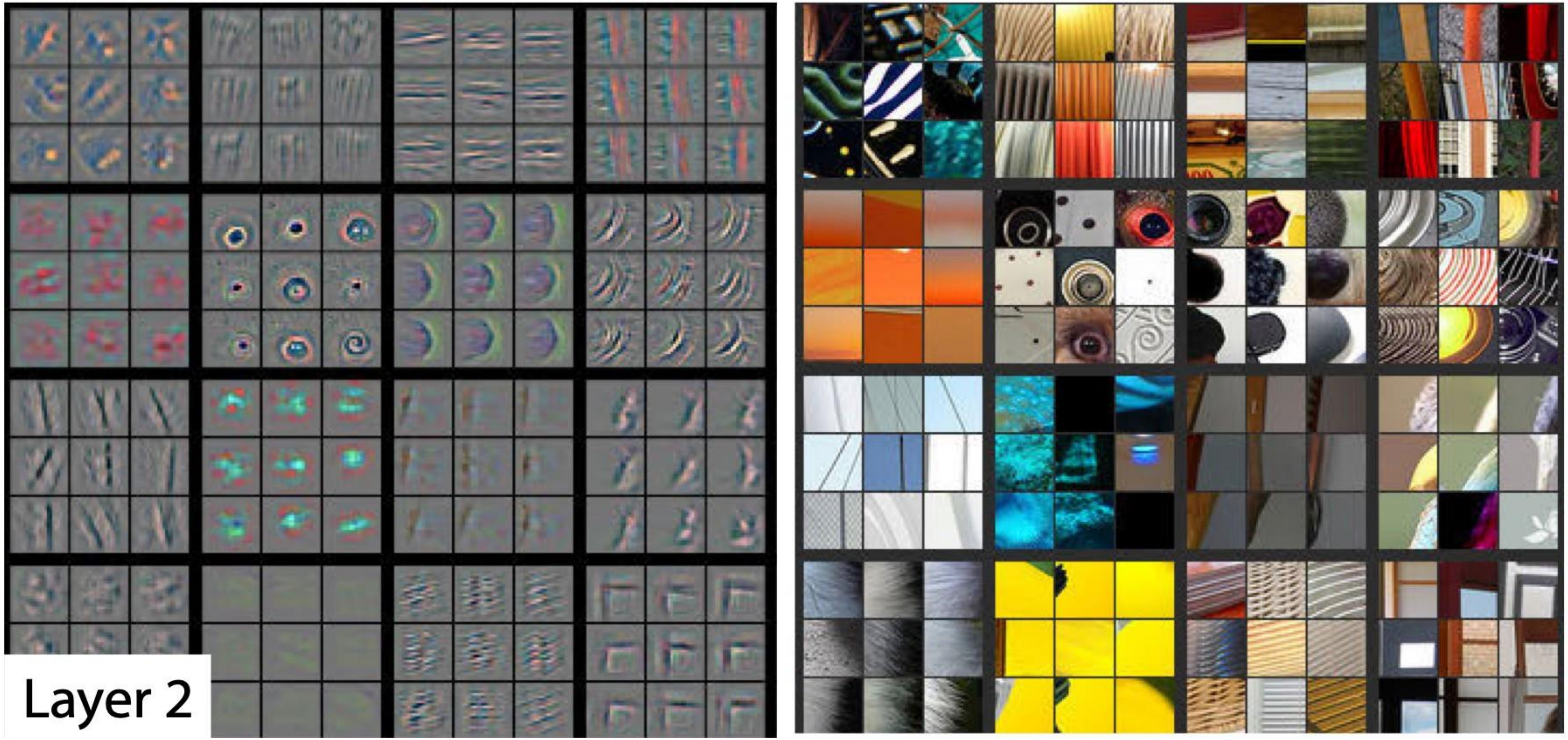
https://arxiv.org/abs/1403.6382

Last layer embeddings



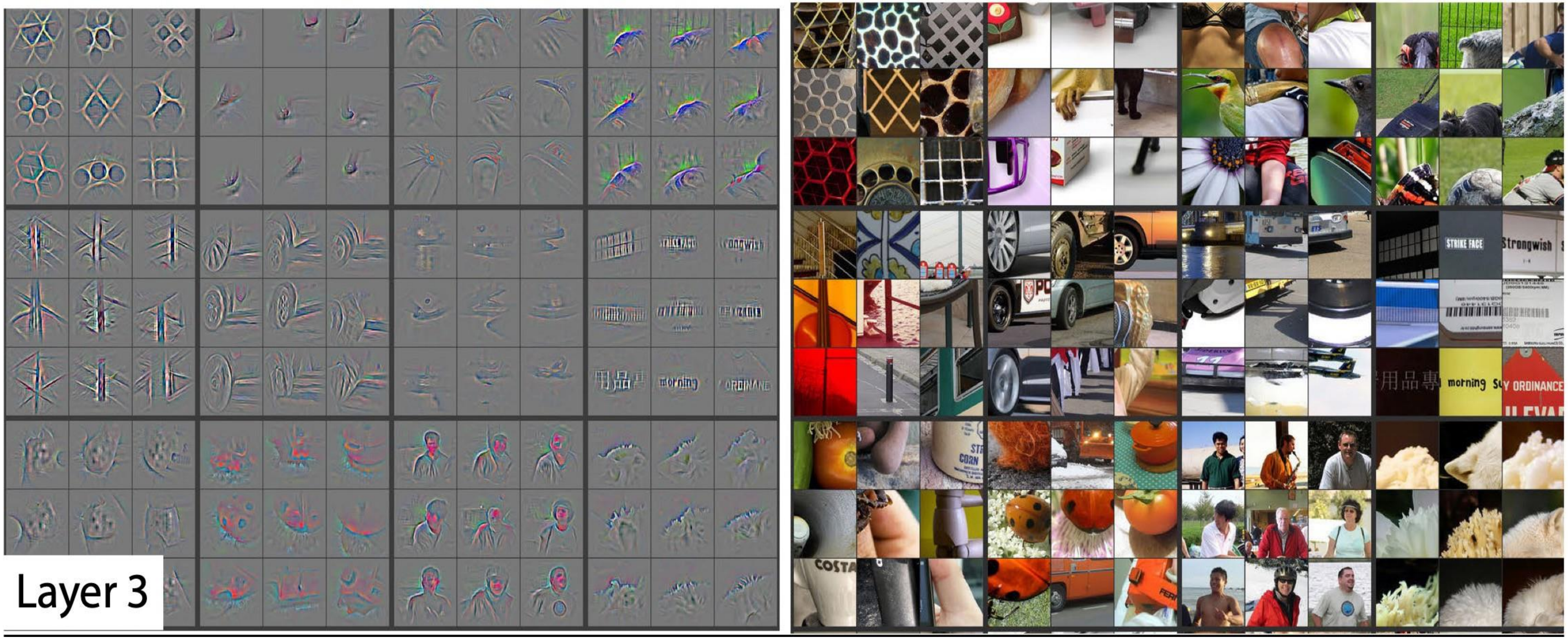
https://arxiv.org/abs/1311.2901

Last layer embeddings



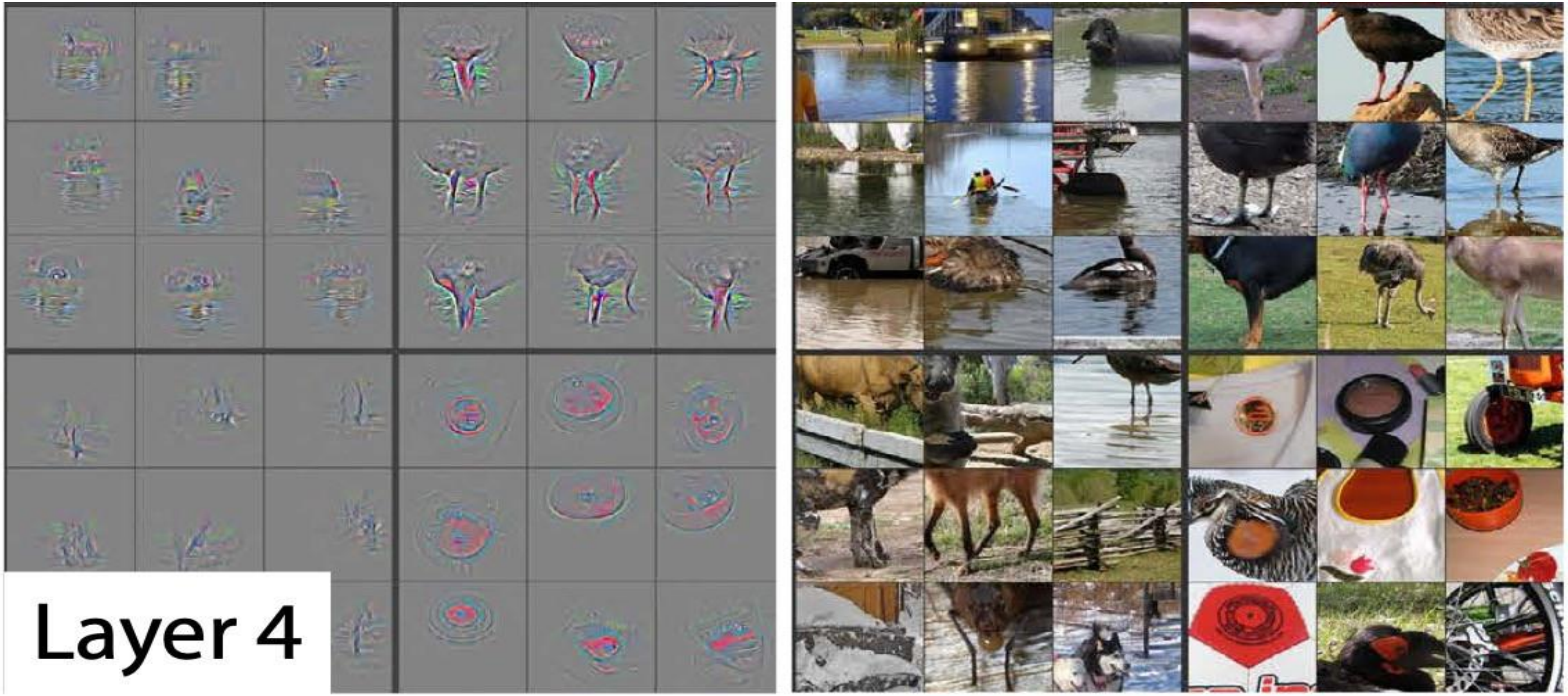
https://arxiv.org/abs/1311.2901

Last layer embeddings



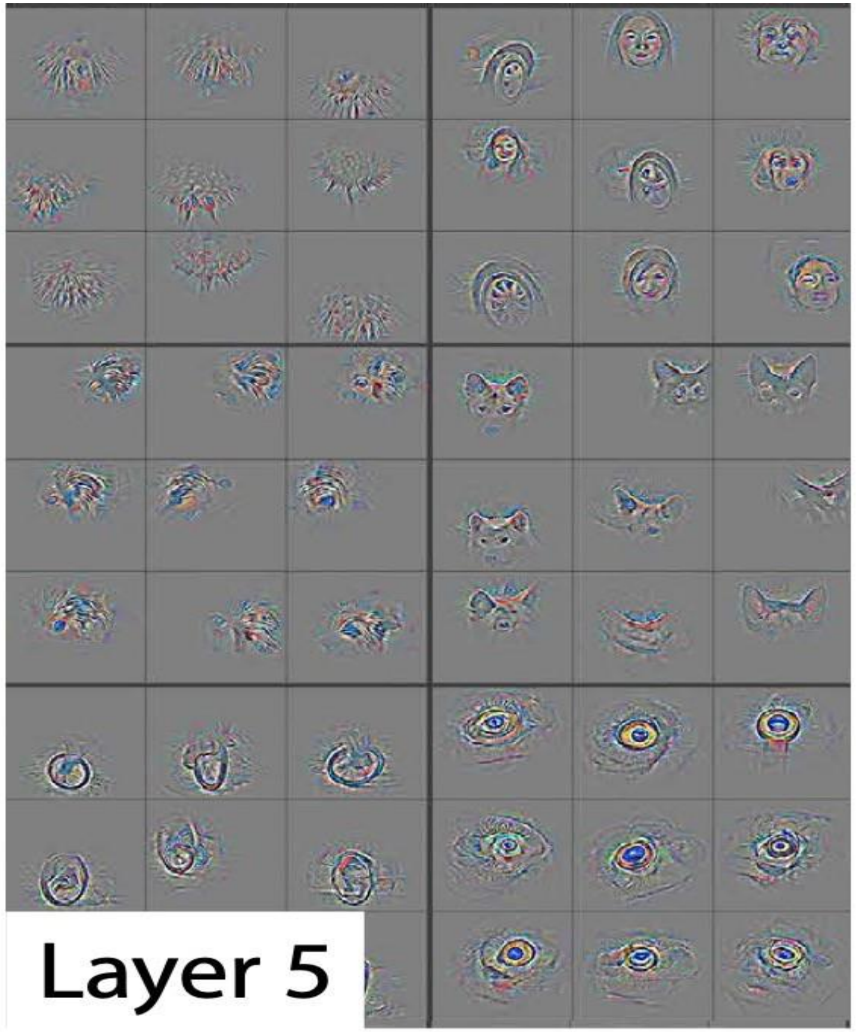
https://arxiv.org/abs/1311.2901

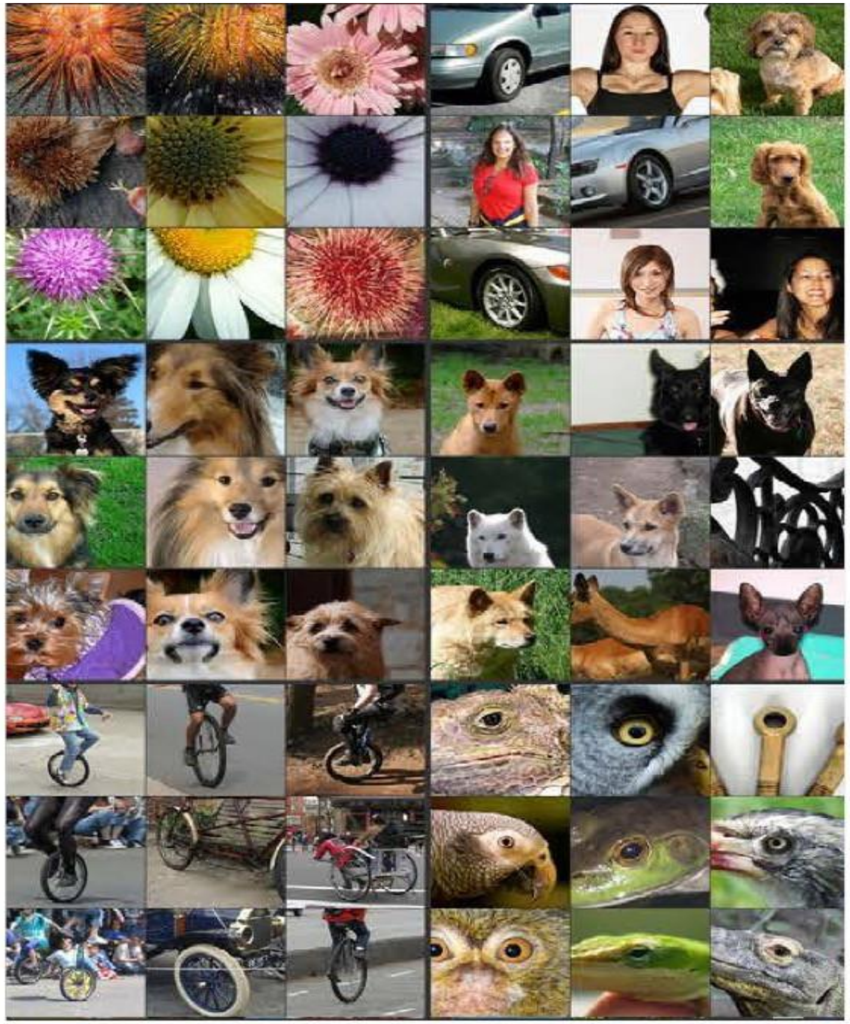
Last layer embeddings



https://arxiv.org/abs/1311.2901

Last layer embeddings





https://arxiv.org/abs/1311.2901